יסודות מערכות פתוחות
פתרון תרגיל מס' 10

שימו לב: כל ההערות שבתחילת תרגילים 1-9 תקפות גם לתרגיל זה.

1. שאלה זו הופיעה במבחן מועד א 2017

בספרית בנימינה שומרים נתונים על ספרים בקובץ בשם books שכל שורה שלו היא במבנה הבא:

תאריך החזרה : תאריך השאלה : שם מחבר : שם ספר : מספר עותק

עבור עותק שלא הושאל אף פעם מופיע – (מינוס) בשדות של תאריך ההשאלה ותאריך ההחזרה.

עבור עותק שהושאל ועדין לא הוחזר מופיע - (מינוס) בתאריך ההחזרה.

עותק מסוים נחשב פנוי אם אין אף שורה בקובץ שמצינת שהעותק הושאל ועדין לא הוחזר.

לדוגמה השורה הבאה:

```
1234:The Hobbit:J. R. R. Tolkien:1/1/2016:20/1/2016
```

מצינת שהעותק מספר 1234 של הספר The Hobbit הושאל בתאריך 1/1/2016 והוחזר בתאריך 20/1/2016

השורה הנ"ל לא בהכרח מצינת שהעותק 1234 פנוי כי יתכן ויש שורה נוספת בקובץ שמצינת שהוא הושאל (לאחר מכן) ועדין לא הוחזר.

השורה הבאה:

```
1233:The Hobbit:J. R. R. Tolkien:8/1/2016:-
```

מצינת שהעותק מספר 1233 של הספר The Hobbit הושאל בתאריך 8/1/2016 ועדין לא הוחזר.

השורה הבאה:

```
1232:The Hobbit:J. R. R. Tolkien:-:-
```

מצינת שהעותק מספר 1232 של הספר The Hobbit נמצא כרגע בספריה ולא הושאל אף פעם.

כתוב/כתבי תכנית ב- awk בשם P10.1 שמקבלת כפרמטר שם סופר ומחזירה את כל שמות כל הספרים של הסופר הזה שיש מהם לפחות עותק אחד פנוי בספריה. על כל שם ספר להופיע פעם אחת בדיוק בפלט ובשורה נפרדת ועל שמות הספרים להיות ממויינים לפי סדר לכסיגוקרפי עולה.
במידה ואין כלל ספרים שעבורם יש עותקים פנויים בספריה עבור סופר זה תופיע ההודעה:

```
No available books for this author
```

מותר להשתמש בפקודת system אחת בלבד לפתרון השאלה.
לדוגמה, נניח שהקובץ books מכיל את הנתונים הבאים:

```
1234:Crime and Punishment:F. Dostoevsky:1/1/2016:20/1/2016
1234:Crime and Punishment:F. Dostoevsky:1/2/2016:-
1232:The Hobbit:J. R. R. Tolkien:1/10/2015:5/11/2015
1232:The Hobbit:J. R. R. Tolkien:10/10/2014:1/1/2015
2111:Crime and Punishment:F. Dostoevsky:20/1/2016:-
2111:Crime and Punishment:F. Dostoevsky:20/3/2015:12/4/2015
1233:The Hobbit:J. R. R. Tolkien:8/1/2016:-
1220:The Gambler:F. Dostoevsky:-:-
2112:The Gambler:F. Dostoevsky:20/1/2016:-
2112:The Gambler:F. Dostoevsky:20/1/2015:20/2/2015
3007:The Shack:William P. Young:10/10/2015:-
1236:The Hobbit:J. R. R. Tolkien:-:-
4111:The Idiot:F. Dostoevsky:20/1/2017:20/2/2017
3001:The Shack:William P. Young:25/1/2016:-
5002:The Brothers Karamazov:F. Dostoevsky:12/1/2017:-
5004:The House of the Dead:F. Dostoevsky:13/1/2017:12/2/2017
```

לאחר הפעלת התכנית על ידי הפקודה:

```
P10.1 "F. Dostoevsky"
```

מתקבל הפלט:

```
The Gambler
The House of the Dead
The Idiot
```

שימו לב שהספרים Crime and Punishment ו- The Brothers Karamazov לא מופיעים בפלט כי אין עותקים פנויים שלהם בספריה.

לאחר הפעלת התכנית על ידי הפקודה:

```
P10.1 "William P. Young"
```

מתקבל הפלט:

```
No available books for this author
```

פתרון שאלה 10.1:

```
#!/bin/awk -f
function get_books_of_author(author,A,aut,book){
  while (getline<"books" == 1){
    split($0,A,":")
    aut=A[3];book=A[2]
    if (aut==author)  print book >"tmp"
  }
  close("books");close("tmp")
}
function get_ids_of_book(book,A,book1,id) {
  while (getline<"books" == 1){
    split($0,A,":")
    book1=A[2];id=A[1]
    if (book==book1)  print id >"tmp1"
  }
  close("books");close("tmp1")
}
function is_id_avail(id,A,id1){
  while (getline<"books" == 1){
    split($0,A,":")
    id1=A[1];borrow_date=A[4];return_date=A[5]
    if (id1 != id) continue
    if (borrow_date != "-" && return_date == "-") {
      close("books"); return "NO"
    }
  }
  close("books"); return "YES"
}
function is_book_avail(book){
  get_ids_of_book(book)
  while (getline<"tmp1" == 1){
    id=$0;
    if (is_id_avail(id)=="YES"){
      close("tmp1"); return "YES"
    }
  }
  close("tmp1"); return "NO"
}
BEGIN {
  author=ARGV[1]
  get_books_of_author(author)
  system("sort -u tmp>|tmp2")
  while (getline<"tmp2" == 1){
    book=$0;
    if (is_book_avail(book)=="YES") print book
  }
}
```

ובפורמט טקסט:

```
#!/bin/awk -f
function get_books_of_author(author,A,aut,book){
  while (getline<"books" == 1){
    split($0,A,":")
     aut=A[3];book=A[2]
     if (aut==author)  print book >"tmp"
  }
  close("books");close("tmp")
}
function get_ids_of_book(book,A,book1,id) {
  while (getline<"books" == 1){
    split($0,A,":")
     book1=A[2];id=A[1]
     if (book==book1)  print id >"tmp1"
  }
  close("books");close("tmp1")
}
function is_id_avail(id,A,id1){
  while (getline<"books" == 1){
    split($0,A,":")
     id1=A[1];borrow_date=A[4];return_date=A[5]
     if (id1 != id) continue
     if (borrow_date != "-" && return_date == "-") {
       close("books"); return "NO"
    }
  }
  close("books"); return "YES"
}
function is_book_avail(book){
  get_ids_of_book(book)
  while (getline<"tmp1" == 1){
    id=$0;
     if (is_id_avail(id)=="YES"){
       close("tmp1"); return "YES"
     }
  }
  close("tmp1"); return "NO"
}
BEGIN {
  author=ARGV[1]
  get_books_of_author(author)
  system("sort -u tmp>|tmp2")
  while (getline<"tmp2" == 1){
    book=$0;
     if (is_book_avail(book)=="YES") print book
  }}
```

2. שאלה זו מבוססת על שאלה שהופיעה במבחן מועד ב 2017

כתוב פונקציה ב- awk במבנה הבא:

```
function is_date_less_than(date1,date2) {

}
```

שמקבלת כפרמטרים שני תאריכים בפורמט של dd/mm/yyyy
ומחזירה YES אם התאריך הראשון קטן מהתאריך השני,
אחרת הפונקציה מחזירה NO.

ניתן להניח שיום מתואר על ידי 1 או 2 ספרות, חודש מתואר על ידי 1 או 2 ספרות ושנה מתוארת על ידי 4 ספרות.
למשל חודש מרץ יכול להיות מתואר על ידי המספר 3 ויכול להיות מתואר גם על ידי המספר 03.

כדי שניתן יהיה לבדוק את הפונציה בתכנית הבדיקה האוטומטית יש לכתוב אותה בתוך קובץ תכנית ב- awk בשם P10.2 שמקבלת כפרמטרים את התאריכים ומדפיסה את התשובה למסך.

במילים אחרות הקובץ P10.2 של התכנית ב- awk נראה כך, ועליכם להשלים רק את החלקים שבתוך הסוגריים המסולסלים של הפונקציה:

```
#!/bin/awk -f
function is_date_less_than(date1,date2) {
...
...
}
BEGIN { print is_date_less_than(ARGV[1],ARGV[2]) }
```

לדוגמה, לאחר הקריאה לתוכנית על ידי הפקודה:

```
P10.2 1/1/2016 01/12/2016
```

מתקבל הפלט:

```
YES
```

לאחר הקריאה לתוכנית על ידי הפקודה:

```
P10.2 1/1/2016 1/1/2016
```

מתקבל הפלט:

```
NO
```

פתרון שאלה 10.2:

```
#!/bin/awk -f
function is_date_less_than(date1,date2,A,d1,d2,m1,m2,y1,y2){
  split(date1,A,"/")
  d1=A[1];m1=A[2];y1=A[3]
  split(date2,A,"/")
  d2=A[1];m2=A[2];y2=A[3]
  if (y1<y2) return "YES"
  if (y1>y2) return "NO"
  if (m1<m2) return "YES"
  if (m1>m2) return "NO"
  if (d1<d2) return "YES"
  return "NO"
}
BEGIN { print is_date_less_than(ARGV[1],ARGV[2]) }
```

ובפורמט טקסט:

```
#!/bin/awk -f
function is_date_less_than(date1,date2,A,d1,d2,m1,m2,y1,y2){
  split(date1,A,"/")
  d1=A[1];m1=A[2];y1=A[3]
  split(date2,A,"/")
  d2=A[1];m2=A[2];y2=A[3]
  if (y1<y2) return "YES"
  if (y1>y2) return "NO"
  if (m1<m2) return "YES"
  if (m1>m2) return "NO"
  if (d1<d2) return "YES"
  return "NO"
}
BEGIN { print is_date_less_than(ARGV[1],ARGV[2]) }
```

3. כתוב תכנית סקריפט ב- bash בשם P10.3 שמקבלת כפרמטרים שני תאריכים ומדפיסה את מספר הימים בין התאריך הראשון לתאריך השני (לא כולל התאריך השני). על הפורמט של הפלט להיות כפי שמוצג בדוגמה שלהלן. על התכנית P10.3 להשתמש בתכנית עזר בשם P10.3.1 ב- expect שניגשת לאתר:

https://www.calculatorsoup.com/calculators/time/time-date-difference-calculator.php

ומחשבת את התוצאה באמצעות האתר הנ"ל.
חיבים להשתמש באתר הנ"ל (כדי לתרגל expect ו- elinks), ולכן תכנית שתבצע את החישוב הנדרש ללא שימוש באתר הנ"ל לא תתקבל.

לדוגמה, לאחר הקריאה לתכנית:

P10.3 4/1/2018 3/2/2018

יתקבל הפלט:

The number of days between 4/1/2018 and 3/2/2018 is:30

לדוגמה, לאחר הקריאה לתכנית:

P10.3 9/1/2018 8/2/2019

יתקבל הפלט:

The number of days between 9/1/2018 and 8/2/2019 is:395

<u>הדרכה:</u>

1. כשמשתמשים באתר הנ"ל שימו לב שהתאריכים שנדרשים באתר הם בפורמט של mm/dd/yy ששונה מהפורמט של התאריכים בקלט לתכנית שהוא: dd/mm/yy

2. כשמשתמשים באתר הנ"ל יש לשים 0:00 בשדות של זמן ההתחלה וזמן הסיום.

3. כשמשתמשים באתר הנ"ל לאחר מילוי שדה אחד, כשרוצים למלא שדה נוסף יש לשלוח חץ תחתון, התו של חץ תחתון מתואר על ידי הרצף:

"\033\[B"

4. כדי שהימים יקחו בחשבון גם שנים וחודשים יש לבקש באתר להראות את התוצאה בכל היחידות, ואז להציג את החלק השלם התחתון של מספר הימים שמתקבל.

פתרון שאלה 10.3:

להלן תוכן התכנית P10.3.1:

```
#!/usr/bin/expect
log_user 0
file delete -force tmp
#set timeout 2
set date1 [lindex $argv 0]
set date2 [lindex $argv 1]
set Alt "\033"
set cu "\025"
set ar "\033\[B"
set time "0:00"
spawn elinks https://www.calculatorsoup.com/calculators/time/time-date-difference-calculator.php
sleep 4
expect "Link" {send "22\n"}
expect "Time" {send "\n$cu$date1$ar\n$cu$time$ar\n$cu$date2$ar\n$cu$time$ar\n$ar\n\n"}
sleep 4
expect "Time" {send "${Alt}fv${cu}tmp\n"}
expect * {send "q"}
expect * {send "y"}
expect
```

להלן תוכן התכנית P10.3:

```
d1=$1
d2=$2
m="$(echo $d1 | cut -d"/" -f2)"
d="$(echo $d1 | cut -d"/" -f1)"
y="$(echo $d1 | cut -d"/" -f3)"
d1="$m/$d/$y"
m="$(echo $d2 | cut -d"/" -f2)"
d="$(echo $d2 | cut -d"/" -f1)"
y="$(echo $d2 | cut -d"/" -f3)"
d2="$m/$d/$y"
P10.3.1 $d1 $d2
dd="$(egrep -o "= [0-9]+ days" tmp | cut -d" " -f2)"
s="The number of days between $1 and $2 is:$dd"
echo $s
```

להלן התכנית P10.3.1 בפורמט טקסט:

```
#!/usr/bin/expect
log_user 0
file delete -force tmp
#set timeout 2
set date1 [lindex $argv 0]
set date2 [lindex $argv 1]
set Alt "\033"
set cu "\025"
set ar "\033\[B"
set time "0:00"
spawn elinks https://www.calculatorsoup.com/calculators/time/time-date-difference-calculator.php
sleep 4
expect "Link" {send "22\n"}
expect "Time" {send "\n$cu$date1$ar\n$cu$time$ar\n$cu$date2$ar\n$cu$time$ar\n$ar\n\n"}
sleep 4
expect "Time" {send "${Alt}fv${cu}tmp\n"}
expect * {send "q"}
expect * {send "y"}
expect
```

להלן התכנית P10.3 בפורמט טקסט:

```
d1=$1
d2=$2
m="$(echo $d1 | cut -d"/" -f2)"
d="$(echo $d1 | cut -d"/" -f1)"
y="$(echo $d1 | cut -d"/" -f3)"
d1="$m/$d/$y"
m="$(echo $d2 | cut -d"/" -f2)"
d="$(echo $d2 | cut -d"/" -f1)"
y="$(echo $d2 | cut -d"/" -f3)"
d2="$m/$d/$y"
P10.3.1 $d1 $d2
dd="$(egrep -o "= [0-9]+ days" tmp | cut -d" " -f2)"
s="The number of days between $1 and $2 is:$dd"
echo $s
```

.4

כתוב תכנית סקריפט ב- **bash** בשם **P10.4** שמקבלת כפרמטר שם עיר ומדפיסה לפלט את השעה בעיר זו (על פורמט הפלט להיות בהתאם למתואר בדוגמה שבהמשך). על התכנית **P10.4** להשתמש בתכנית עזר בשם **P10.4.1** ב- **expect** שניגשת לאתר:

http://www.checktimes.com/

בדומה לדוגמה 9 של הרצאה 10.
התכנית שלכם צריכה לשפר את התכנית שהוצגה באופן הבא:

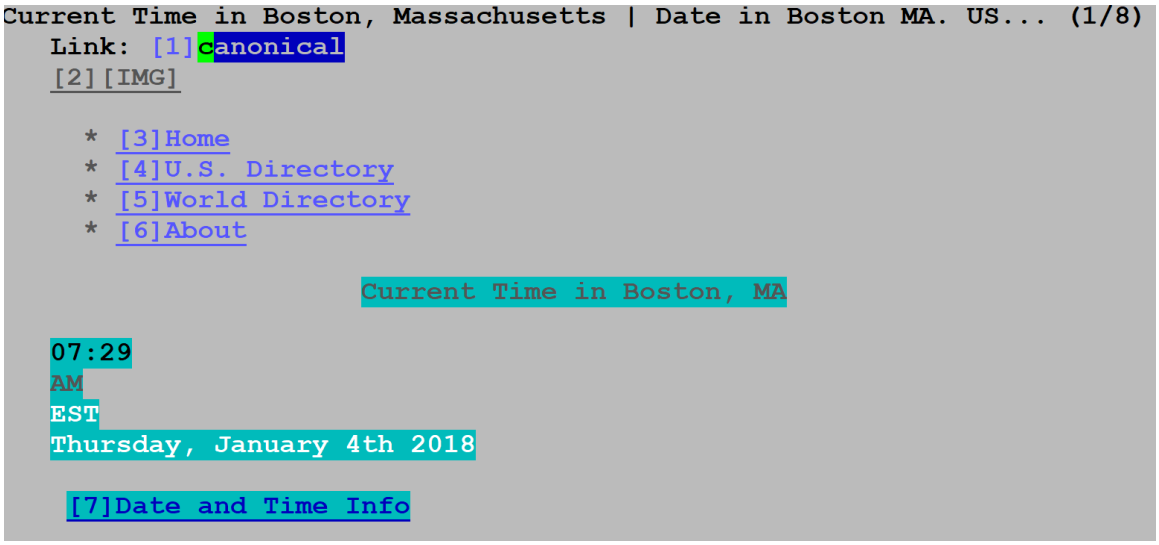
אם לעיר שהתבקש עבורה חיפוש יש יותר מתוצאת חיפוש אחת, על התכנית להציג את התוצאה עבור העיר הראשונה מבין רשימת הערים שעונות לחיפוש.

לדוגמה אם נגלוש באתר הנ"ל באמצעות **elinks** ונחפש את **boston** נקבל את המסך הבא:

```
  [1][IMG]

    * [2]Home
    * [3]U.S. Directory
    * [4]World Directory
    * [5]About

esults of Search for "boston"

  ------------------------------------------------------------------------

    * [6]Boston, Massachusetts, United States of America
    * [7]Boston, England, United Kingdom
    * [8]Boston Spa, England, United Kingdom
    * [9]New Boston, New Hampshire, United States of America
    * [10]New Boston, Texas, United States of America
    * [11]Boston, Davao, Philippines
```

במצב הזה נרצה להמשיך ולגלוש כדי לקבל את התשובה המתאימה לאופציה הראשונה שמתוארת בשדה 6 במסך הנ"ל.

לאחר הגלישה לאופציה בשדה 6 נקבל את המסך הבא:



לכן לאחר הפעלת התכנית:

```
P10.4 boston
```

יתקבל הפלט:

```
The current time in Boston, MA is: 07:29
```

שימו לב שהשעה שתודפס תלויה בזמן שבו הפעלתם את התכנית.

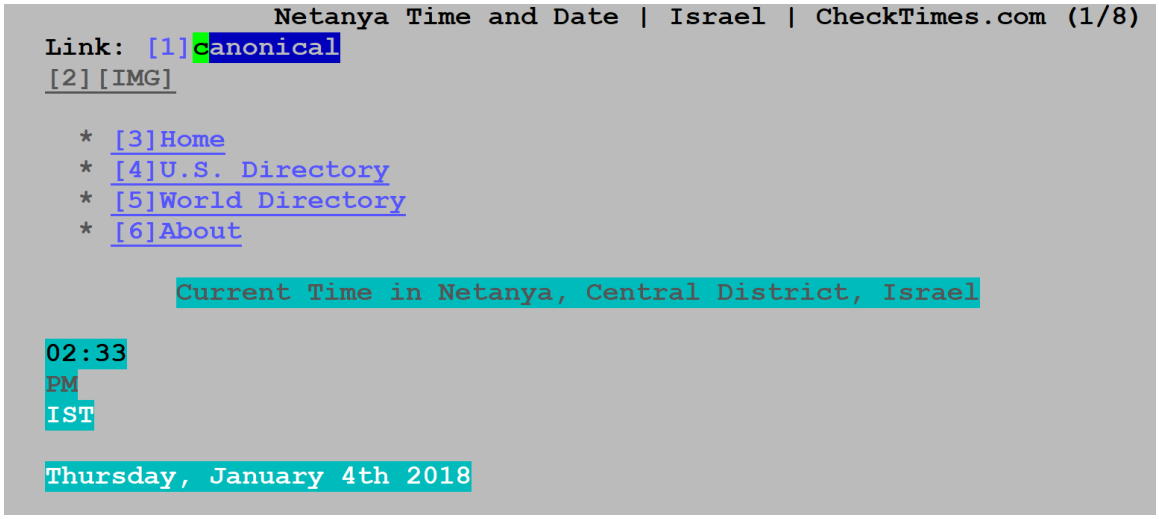
מצד שני על התכנית לעבוד נכון גם עבור ערים שיש להן רק אופציה אחת. ולכן לאחר הפעלת התכנית על ידי הפקודה:

```
P10.4 netanya
```

יתקבל הפלט:

```
The current time in Netanya, Central Discrict, Israel is: 02:33
```

שימו לב שהתאור Netanya, Central Discrict, Israel התקבל לאחר החיפוש של netanya באתר כפי שמתואר במסך הבא:



<u>הדרכה</u>

כדי להבדיל בין שני המקרים אפשר להשתמש במבנה הבא של הפקודה expect:

```
expect {
    string1   { program 1 }
    string2   { program 2}
    …
}
```

משמעות המבנה הנ"ל הוא שאם יש התאמה ל- string1 תבוצע program1 ואם יש התאמה ל- string2 תבוצע program2. ההתאמה הראשונה מבין השניים תבוצע. (כך שאו ש- program1 תבוצע או ש- program2 תבוצע, או שאף אחת מהן לא תבוצע אם יגיע ה- timeout). אבל אף פעם לא תבוצע גם program1 וגם program2.

סדר הבדיקה הוא: קודם בודקים את string1 ואחר כך בודקים את string2 וחוזר חלילה עד שיש התאמה או שמגיע timeout.

פתרון שאלה 10.4:

להלן תוכן התכנית P10.4.1:

```
#!/usr/bin/expect
log_user 0
file delete -force tmp
#set timeout 2
set city [lindex $argv 0]
set Alt "\033"
set Ctrl_u "\025"
spawn elinks http://www.checktimes.com/
sleep 4
expect "Current" {send "9\n"}
expect "Value" {send "\n$Ctrl_u$city\n"}
expect "form" {send "\n"}
sleep 4
expect {
 "Current" {
    expect "Time" {send "${Alt}fvtmp\n"}
    expect * {send "q"}
    expect * {send "y"}
    expect
 }
 "Results" {
    expect * {send "6\n\n"}
    sleep 4
    expect "Time" {send "${Alt}fv${Ctrl_u}tmp\n"}
    expect * {send "q"}
    expect * {send "y"}
    expect
 }
}
```

להלן תוכן התכנית P10.4:

```
city=$1
P10.4.1 $city
s1="$(egrep -o "in [A-Za-z, ]+" tmp | head -1)"
t1="$(egrep -o "[0-9][0-9]:[0-9][0-9]" tmp | head -1)"
echo "The current time in $s1 is: $t1"
```

התכנית P10.4.1 בפורמט טקסט:

```
#!/usr/bin/expect
log_user 0
file delete -force tmp
#set timeout 2
set city [lindex $argv 0]
set Alt "\033"
set Ctrl_u "\025"
spawn elinks http://www.checktimes.com/
sleep 4
expect "Current" {send "9\n"}
expect "Value" {send "\n$Ctrl_u$city\n"}
expect "form" {send "\n"}
sleep 4
expect {
 "Current" {
    expect "Time" {send "${Alt}fvtmp\n"}
    expect * {send "q"}
    expect * {send "y"}
    expect
 }
 "Results" {
    expect * {send "6\n\n"}
    sleep 4
    expect "Time" {send "${Alt}fv${Ctrl_u}tmp\n"}
    expect * {send "q"}
    expect * {send "y"}
    expect
 }
}
```

התכנית P10.4 בפורמט טקסט:

```
city=$1
P10.4.1 $city
s1="$(egrep -o "in [A-Za-z, ]+" tmp | head -1)"
t1="$(egrep -o "[0-9][0-9]:[0-9][0-9]" tmp | head -1)"
echo "The current time in $s1 is: $t1"
```